Petteri Streng

Software Manufacturing

Case Study Hammerkit

Helsinki Metropolia University of Applied Sciences
Engineer
Industrial Management
Thesis
08.05.2012

Metropolia

Tiivistelmä

Author Title Number of Pages Date	Petteri Streng Software Manufacturing 59 pages + 1 appendices 8 May 2012
Degree	Engineer
Degree Programme	Industrial Management
Specialisation option	Global Information and Communication Technology
Instructors	Anu Halme, Mark Sorsa-Leslie, Contact Persons Nina Hellman, Lecturer

Nowadays customers are demanding more customized products, better service and lower costs. In order to keep up with these increased demands, companies have to improve their processes continuously. Hammerkit is a small company that manufactures software for other companies. They focus on large Public Relations agencies. Hammerkit is responding to increased customer demands by offering new products through mass customization. To cope with mass customization in their company they are using lean methods to manufacture their products.

The main objective of this thesis was to explore the software manufacturing methods at Hammerkit. The idea was to have an objective point of view and to see how the methods were implemented. This thesis focused on looking at the processes through what is known as lean methods. Hammerkit was already using lean methods in their manufacturing so this was natural. Testing and programing were left out to keep the scope of the thesis reasonable.

The theoretical framework was built on a discussion on lean methods, mass customization and the common problems found in software manufacturing. The focus was mainly on the processes themselves as well as analyzing whether the biggest mistakes in manufacturing could be avoided.

The empirical part focused on Hammerkits' way of working and whether the methods were used in every day production. The question was were they used to their full potential and if not, could they be improved The execution of the manufacturing process was assessed based on the writer's own experience, two interviews with the company representatives and Hammerkit's documents.

The conclusion highlights the most important findings of the thesis with an emphasis on what was found, what was expected and the writer's own opinions as well as recommendations for the future.

Keywords	software, manufacturing, lean, mass customization

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Petteri Streng Ohjelmistojen Valmistus 59 sivua + 1 liite 8 Toukokuuta 2012
Tutkinto	Insinööri
Koulutusohjelma	Tuotantotalous
Suuntautumisvaihtoehto	Globaali ICT
Ohjaajat	Anu Halme, Mark Sorsa-Leslie, Yhteyshenkilöt Nina Hellman, Lehtori

Nykyään asiakkaat vaativat yhä yksilöllisöidyimpiä tuotteita, parempaa palvelua ja alempia hintoja. Yritysten on jatkuvasti parannettava prosessejaan vastatakseen näihin kasvaneisiin vaatimuksiin. Hammerkit on pieni yhtiö, joka tuottaa ohjelmistoja muille yrityksille. He keskittyvät suuriin PR yrityksiin. Hammerkit vastaa kasvaneisiin vaatimuksiin tarjoamalla uusia tuotteita massakustomoinnin avulla. Massakustomoinnin avuksi yritys käyttää lean metodeja ohjelmistotuotannossa.

Tämän opinnäytetyön päätavoite oli tutkia ohjelmistotuotannon metodeja Hammerkitillä. Tavoitteena oli saada objektiivinen näkökulma ja katsoa miten prosessit toimivat. Tämä opinnäytetyö tutki prosesseja lean metodien näkökulmasta. Hammerkit käytti jo ennalta näitä metodeja, joten päätös oli luonnollinen. Testaus ja ohjelmointi jätettiin pois työstä, jotta työn laajuus olisi kohtuullinen.

Teoriamalli pohjustettiin lean metodeihin, massakustomointiin ja yleisiin ongelmakohtiin ohjelmistovalmistuksessa. Pääpaino oli prosesseissa itsessään. Tämän lisäksi tutkittiin voitaisiinko yleisimmät ongelmat ohjelmistotuotannossa välttää.

Empiirinen osa keskittyi Hammerkitin työtapoihin ja miten prosesseja käytettiin työnteossa päivittäin. Kysymyksenä oli, käytettiinkö metodeja parhaalla mahdollisella tavalla ja jos ei, niin voitaisiinko niitä kehittää. Prosessien toimivuus arvioitiin kirjoittajan oman kokemuksen, kahden haastattelun ja Hammerkitin dokumenttien pohjalta.

Päätelmässä otetaan esille tärkeimmät löydökset opinnäytetyöstä painotuksena mitä löytyi, mitä oli odotettavissa ja kirjoittajan omat mielipiteet ja suositukset tulevaisuuden varalle.

Avainsanat	ohjelmisto, valmistus, lean, massakustomointi

Contents

1 Introduction

This section will illustrate the research question and the goal of this thesis. It will explore the execution of the thesis as well as give an insight to the company Hammerkit.

1.1 Business Problem and Research Question

There are different ways to produce software. The problem that companies face is which methodology and practices to use. There is no one best way to produce software, but rather the best way for that company at that time.

The research question of this paper is to look at **how the software manufacturing processes work at Hammerkit and how they could be improved**. Hammerkit is a company that produces business-to-business software. Their main focus is on multinational Public Relations agencies.

This thesis takes a look at the manufacturing methods used at Hammerkit at the moment. Testing and the actual code that goes into the software will not be discussed in detail in order to keep the scope of the thesis within the given limits. This thesis focuses on the processes around coding. Testing is closely related to coding and therefore will not be discussed either. This was decided by the author to keep the array of theories as well as the length of the thesis reasonable.

1.2 Research Methodology

This section is divided into three different categories concerning the thesis; theory, information on Hammerkit and the execution of the thesis.

Theory

The thesis will be completed by studying books and internet databases on software manufacturing, software design and mass customization. The main focus will be on lean methods, mass customization and common problems. This study will also look into

the architecture of software projects. The broad base of different topics is necessary since this subject area is wide. The main aim of this thesis is to identify the best patterns that the most successful software manufacturing companies and projects have in common and to assess whether Hammerkit could adopt some of these to enhance their projects. Another objective is to detect the most common mistakes that take place during software projects and know how to avoid them.

The theoretical part of this thesis focuses on three main parts:

- Lean methods
- Mass customization
- Common problems

Hammerkit uses lean methods and therefore it was given that lean methods will be studied. Hammerkit is focusing on Public Relations agencies. These agencies are best served with mass customization and it is an important part of the new business direction Hammerkit wishes to develop. Common problems were included into the thesis to help alleviate the problems that most of the software projects and companies face.

Information on Hammerkit

The information on Hammerkit for this thesis was gathered via three different methods; written documents, interviews and personal experience. The documents were provided by Hammerkit. Some were public documents, others were for employees only. Two of the documents were internal and concerned the sales processes and the way of manufacturing sites. The third document was used for promoting the cloud store and was for external use. Personal information was gathered through a school project as well as through the author's own personal working experience. The interviews were not carried out in a formal way. The goal of the interviews was to give a better view of what the thesis should be about. The conversations were about concepts relevant to the thesis and what the scope of the thesis should be. The information gathered was written down by the author.

This study is therefore based on the following material:

- Interview with Mark Sorsa-Leslie, CEO

- Interview with Anu Halme, Sales Director
- Own working experience
- School project for Hammerkit
- 2 internal documents
- 1 external document

The information used on this thesis has been checked by a representative from Hammerkit to make sure the information is valid and can be seen by everyone. There is always the possibility that the information collected is outdated, misunderstood or there is something missing from it. The validation was used to avoid these problems.

Execution of the Thesis

The research starts by looking at theories on software manufacturing, software design and mass customization. The theories studied were decided based on the research question. The goal was to get a broad view on the subject of software manufacturing from books and internet databases. Various sources were used to ensure the reliability of the information. The main focus was given to lean manufacturing methods. The most common problems were detected by searching a lot of information on software manufacturing. The problems that came to the surface again and again were selected for this thesis. There are many more problems of course, but it would not serve the purpose of this research to include all of them.

Figure 1 below illustrates how the thesis was conducted.

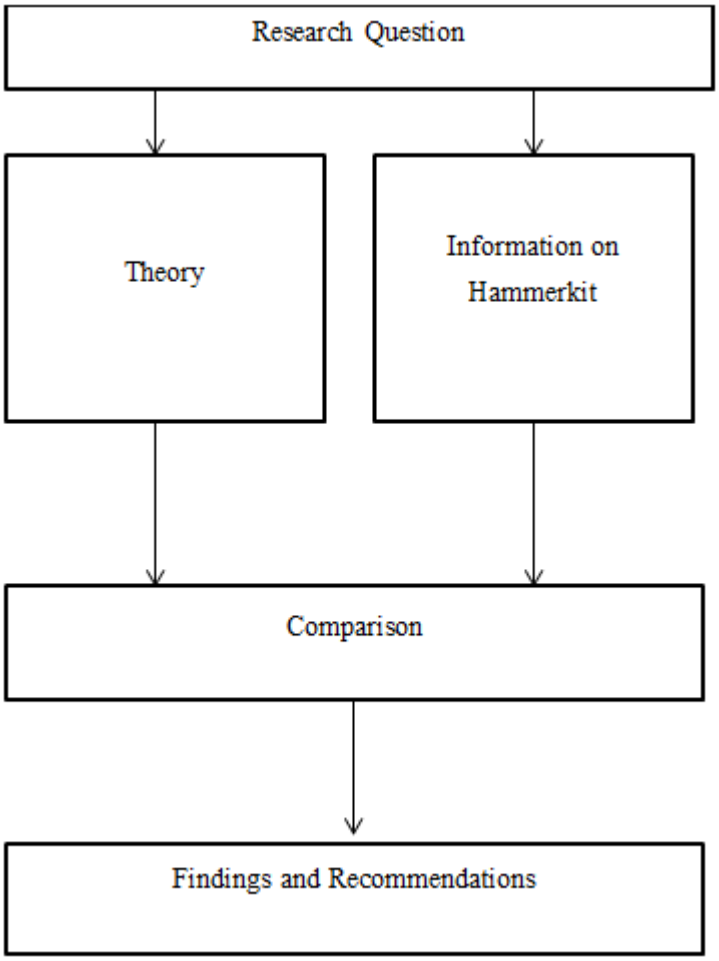

Figure 1. The execution of the thesis.

The theories that were found to be reliable and relevant to the thesis were written down to form the basis for the research on the methods at Hammerkit. Similarly, the documents and other information on Hammerkit were studied for what was considered significant from the point of view of this research.

The theories were reviewed to come up with a model for comparison. The model is a questionnaire that compares the findings from Hammerkit with the theory, in order to find any divergences. This model is presented in chapter 5.

The information on Hammerkit and the theories were then compared to each other. The comparisons were made on the basis of what the author considered to be the best ways for Hammerkit based on the theories. The comparisons provided useful insight for making recommendations for Hammerkit. The comparisons are indicated under

Findings in sections in chapter 6. The findings formed the basis for comparing Hammerkit with the aforementioned table of questions. This was to crystalize all the findings. The recommendations were based on this comparison and are illustrated in chapter 7.

1.3 Hammerkit

Hammerkit was founded in an effort to fight the status-quo in software manufacturing by creating a different way to manufacture and maintain data-driven sites. It began as a company of two coders and two designers in 2006 in Helsinki. At the moment Hammerkit is a company with around 20 employees and the main office is located in Ruoholahti, Helsinki with a sales office in Liverpool in the UK. The goal of Hammerkit was to make designer-friendly tools for creating dynamic web services. Hammerkit 3.5 was launched in 2007. It introduced a different way of building interactive web services. It is offered as a Software as a service (Saas) model which enables Hammerkit to offer a complete online web design and assembly service to its customers. Hammerkit 3.5 includes all the tools that professional users need to create and launch their web products even faster. Hammerkit also provides various different tools to design web pages as well as managing already existing web projects. (http://www.Hammerkit.com/index/43).

The customer feedback demonstrates that the company's service is highly respected. During their short history, the company has made a name for themselves both domestically and internationally. They have already received several prizes and recognitions during the last couple of years. In October 2010 Hammerkit won the Audience Favorite and Best Company awards at the International MindTrek conference in Tampere, Finland. At the end of the same year, Hammerkit was selected as one among ten companies nominated as Young Innovative Company with high-growth potential by the Finnish National Innovation Organization (TEKES). In 2009 Hammerkit received the Red Herring Global 100 Award and also got recognized as one of the 20 most potential European growth companies at the European Venture Forum in Dusseldorf, Germany. These awards have encouraged Hammerkit in their goal set out to become the next generation web service design infrastructure provider. (http://www.Hammerkit.com/index/43).

Hammerkit is now moving to a new direction in software manufacturing. They are focusing on business-to-business sales. This is done via a corporate application store called *the cloud store*. It is a new software platform that is intended for multinational public relations agencies. These agencies can have up to 80 or more offices around the world. The cloud store will provide the company a platform from which the different offices can purchase formats. These formats are generic in the application store. One format is a web site. The office that buys this particular format, will then add their information on it. The goal is to take these generic formats as far as possible so the customer does not have to do a lot of additional work. By building formats for the customers, Hammerkit does not have to manufacture a new one from scratch every time. A particular format can be used in different offices, which can then further modify it to fit their needs. This thesis looks in detail at the software manufacturing processes of cloud store offerings. (http://www.hammerkit.com/services)

2 Mass Customization

At the end of 1980's people started feeling that mass marketing was no longer the way to go. Customers were demanding more customization, they wanted products faster and they wanted better quality. A research area called mass customization was developed in order to bring customers what they demanded. (Pine II 1992: 45. Idsoe, Skjevdal: 1).

Mass customization was introduced as a term by Stan Davis in 1987 as

> "the situation when the same large number of customers can be reached as in mass markets of the industrial economy, and simultaneously they can be treated individually as in the customized markets of pre-industrial economies"
> Davis S.M., 1987, Future Perfect, Addison-Wesley, New York (Idsoe, Skjevdal:1)

There are eight (8) different levels of customization that range from mass production at one end to one-of-a-kind production at the other. Mass customization places in between these two, as can be seen in figure 2 below. (Idsoe, Skjevdal: 2).

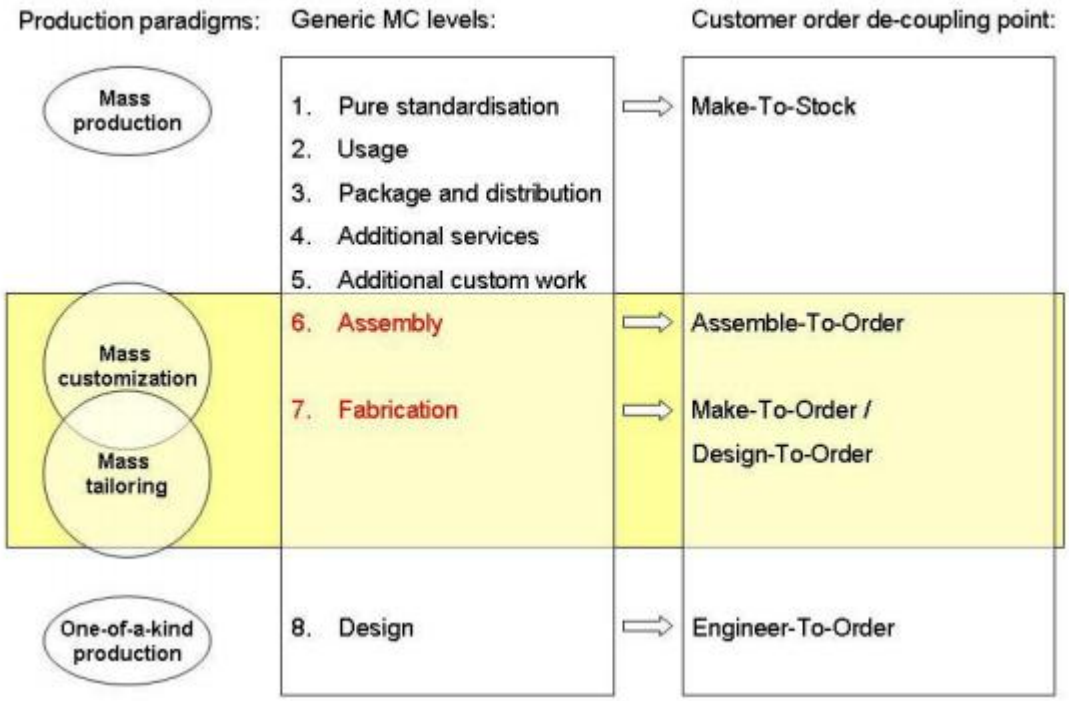

Figure 2. The different production paradigms. (Idsoe, Skjevdal:2)

Usually only levels six and seven are considered mass customization. Level one is only to produce a large quantity of products but it does not offer specific customization for customers. Level eight on the other hand does not produce in bulk, but rather just small quantities for customers as per requested. Levels two to five do not offer a lot of customization for the customer and as such are usually not considered part of mass customization.(Idsoe, Skjevdal:1).

Mass customization uses a model where a product is first designed, then sold and finally made, as opposed to the usual model where products are made before selling them. Mass customization is concerned directly with the end customer. (Badurdeen, Stump 2012: 111).

Mass customization allows a company to customize the products for their customers. This will give a competitive advantage over other companies (Highsmith 2000: 3).
Mass customization switches the demand for the products. The demand for a certain particular product decreases. The overall demand for products increases however. The company will be able to offer more products and the cumulative quantity of niche products is likely to grow. (Pine II 1992: 46).

	Mass Production	Mass Customization
Focus	Efficiency through stability and control	Variety and customization through flexibility and quick responsiveness
Goal	Developing, producing, marketing, and delivering goods and services at prices low enough that nearly everyone can afford them	Developing, producing, marketing, and delivering affordable goods and services with enough variety and customization that nearly everyone finds exactly what they want
Key Features	• Stable demand • Large, homogeneous markets • Low-cost, consistent quality, standardized goods and services • Long product development cycles • Long product life cycles	• Fragmented demand • Heterogeneous niches • Low-cost, high-quality, customized goods and services • Short product development cycles • Short product life cycles

Table 1. The difference between mass production and mass customization. (Pine II 1992: 47).

Table 1 shows the differences in focus, goal and key features between mass production and mass customization. The focus of mass customization is to have the variety and customization needed to be able to offer individual products to customers. The goal is to offer these products at an affordable price. This way a company can offer customized products that the customers want and at the same time keep the costs low. Mass customization focuses therefore on a variety of different niches, or market shares. Product development and product life cycles are shortened with mass customization. (Pine II 1992: 47).

The goal is to make a core product that is developed as far as possible while not customizing it to a specific customer. This enables a company to offer software a lot faster than if it were to build it from scratch for each new project. It also makes the products more customized than if they were just be made as mass production. Mass customization is about getting the best of both worlds. A single process can produce a variety of end products that are similar but customized in the end to a specific customer. This way the company has a broader scope in products. The lowered production costs are a

result of this and this can be transferred to lowered prices for customers. (Pine II 1992: 48).

3 Lean Architecture

Software manufacturing is more than just the coding of everything at the end. The process starts from an idea or a request for a new software or new functionality. This section examines lean methods as a way of organizing this whole process.

3.1 Overview of Lean Software Development

Lean manufacturing was developed by Toyota with the main purpose of eliminating waste, encouraging employees to inform about problems and suggesting ways to fix them and to reduce inventory. (Badurdeen, Stump 2012: 110). Below are highlighted the four main points found again and again in the literature on lean methods. These are not methods but something that needs to be focused on at all times. These points are: *Focus on people, Framework, Code as documentation* and *Always add documentation.*

Focus on people
Focusing on people makes sure that everyone who is involved in the process contribute in some way to it. Everyone has a role to fulfill in lean method. Everyone will feel responsible for the results since they are involved in the project from the beginning to the end. It will also help prevent problems in the future by having different points of view early on. Having everyone involved in the beginning will reduce the costs of the project as well as improve its quality. (Bjørnvig, Coplien 2010: 2-4).

People are the most important part of the project, not the processes. A lot of companies acknowledge this fact but it is not shown in their manufacturing. A company should always put most of their focus on the people. (Highsmith 2000: 100).

The focus should be on people instead of tools. Lean methods put the experience of the people involved before focusing on engineering methods. By giving this freedom to people involved, lean methods also demand them to put in a lot of effort and to work together in order to finish the project in time. (Bjørnvig, Coplien 2010: 5).

Lean manufacturing is focusing on the customer expectations and not what the project team thinks the software should accomplish. Lean methods should be teamwork which is guided by everyone involved. (Bjørnvig, Coplien 2010: 36-37).

Framework

Lean architecture builds a foundation on which agile software development can be executed. It gives the team the freedom to be more creative. Everyone involved knows the goal but it is not set in stone as to how to get there. (Bjørnvig, Coplien 2010: 5).

It does not mean a complete freedom to do whatever people want but rather a permission to focus on adapting to what actually is working. This means not to be bound by strict policies but to have the agility of a small company. (Bjørnvig, Coplien 2010: 5).

Software manufacturing cannot be seen as a mature field and it is changing rapidly with regard to markets and technology as well as demand (Condon 2002; 31). A company that is highly concerned with bureaucracy and documentation will not be able to react to these changes quickly. A company that is small or acts like a small company can bypass some of these time consuming steps. A project team that has the ability to react to sudden imposed demands can recover for example from changes or changing market demands a lot faster.

The general guidelines in lean production give the project team more freedom to do their work and foster emergence. Strict working policies do not work well with information workers. The goal of having general guidelines as to how to do all the work is a lot better than no guidelines at all or having everything planned in advance. (Highsmith III 2000: 200, Highsmith 2000: 33).

Agile software development means that the project can react and adapt to changing situations quickly. This can mean for example a change in the requirements, schedule or reacting to a bug. It means that the team does not put as much weight on the doc-

umentation and bureaucracy as it does to solving the problem at hand. (Gwaltney Jr., Richardson 2005: 11).

Code as documentation

Lean software manufacturing focuses on good code. A good code can and should serve in a way as documentation. A good code can reveal what is happening and how it is happening. The people involved in the process can explain why it is happening. Code however cannot be the only form of documentation even though it is an important part of it. Additional documentation should be used to broaden the view that the code gives. A company should focus on documenting the parts that are not prone to changes. The parts that are most likely going to change, should be made so that they require very little decoding. (Bjørnvig, Coplien 2010: 16, 18).

A company should not put too much weight on documentation. Documentation should be used as a tool. Documentation can be used to help the project reach its goal. It should not be a goal on its own. (Highsmith 2000: 124).

Lean methods do not encourage a heavy documentation process. Information technology is a field in which it is not uncommon to see a lot of documentation of which a major part is never used again. To reiterate, it is one of the main goals of lean methods to strip away all the unnecessary steps and excess documentation. Documentation is in no way taken out completely but it is reduced to a bare minimum. Eliminating excess paper work will also enable for more face-to-face time which is one of the goals in lean. (Bjørnvig, Coplien 2010; 6).

Always add value

Lean architecture is a way of organizing software manufacturing processes. It focuses on adding value to the process on each step along the way. It strips away all the unnecessary actions to keep the process as simple and effective as possible. (Bjørnvig, Coplien 2010: 2-4).

Companies should always remember that their software should bring value to the customers and it should be the main aim (Highsmith 2000: 58). By focusing on adding

value all the time, the lean method takes the customer into consideration throughout the project. It is essential and keeps the process from leading to the wrong trail. (Bjørnvig, Coplien 2010: 2-4).

Eliminating all waste is the first fundamental key in lean production. Every phase should produce value to the customer. (Bjørnvig, Coplien 2010: 39).

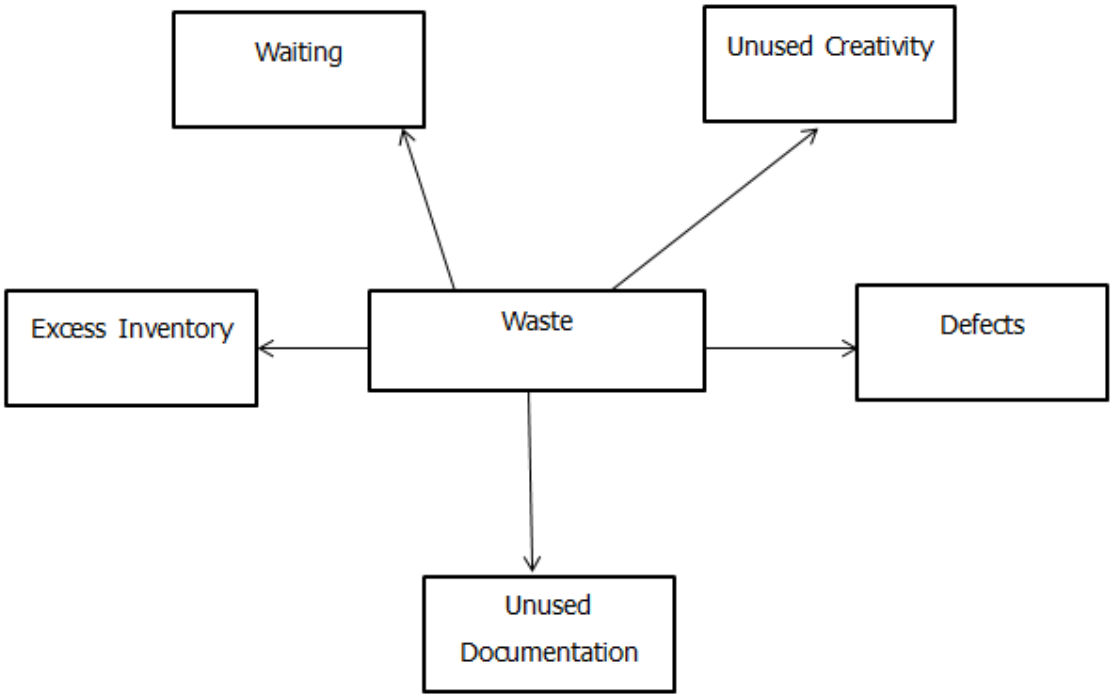

Figure 3. Unnecessary waste in production.

Figure 3 shows the different kinds of waste that should be minimized with lean methods.

There are some specific points to remember when working with lean methods. The first is to avoid producing anything that does not bring value to the end-users. This means documentation as well as code. This is to keep it as lean as possible and to create true value. By stripping away the unnecessary steps it reduces the amount of documentation which will shorten the lifecycle. (Bjørnvig, Coplien 2010: 5, 132).

The software manufacturing processes and methods should be structured on value increasing steps instead of different manufacturing steps. The next point is to create all

the environments and processes in ways that focus on reducing rework. Rework will take time and resources and does not bring any value. The process should flow continuously from the beginning to the end. (Bjørnvig, Coplien 2010: 36-37, 132).

3.2 Stakeholders

This section is divided in two main aspects concerning stakeholders in a software manufacturing project.

Everyone together from the beginning

Everyone who is connected to the project is considered a stakeholder in lean production. This includes the management, programmers, customers and testers and more. The more people are involved the more responsibility they will take for the project. By engaging everyone in the project early on will give them a possibility to influence the outcome. Other stakeholders will also know who is mainly responsible for a certain part of the project. (Highsmith III 2000: 131, Bjørnvig, Coplien 2010: 27).

A system is after all a combination of different parts, and by engaging people with knowledge from different parts of the system will benefit the final product. This is best done early on, as any problem detected at this point will be much easier to fix or change than later on. It will also help to determine the required functions of the software. (Condon 2002: 78, Bjørnvig, Coplien 2010: 28).

There will be a lot more viewpoints when all the stakeholders are together. Determining the specific situations for a certain project should always be a group activity. This way the possibility of encountering a situation that has not been prepared for can be reduced. These situations can be caused by inaccurate programming, invalid systems or user mistakes. Therefore it is important to have people with different backgrounds to brainstorm these situations. (Schlesinger 2010; 28, 37).

The customer needs to be listened to in the beginning as well as throughout the project (Highsmith 2000: 61). Involving everyone from the beginning to the end will help

with defining the problem. It will more likely mirror what the customers want from the software. (Krishnamurthy, Saran 2008: 62).

Good communication

It is important that different stakeholders understand where the others stand on a project and what they want or need (Condon 2002; 21). Any successful project needs different points of view that bring value or reduce the risks of the project. It is however a problem if there is not a mutual understanding amongst the stakeholders or if people do not listen to one another. That problem should be eliminated or at least reduced if there is trust between the members of the team (Condon 2002; 22).

Successful projects always have a team that has a good collaboration. Regardless of the size of the project, there will always be communication between coders, designers, customer, etc. It should be a key priority to make sure the collaboration and communication is working fluently in any project. Regular communications between the whole team is what actually builds the team from just people who are working at the same project. It is clear that getting everyone together and having them interact with each other will be of enormous benefit for the whole project. Meetings allow employees to help each other. (Gwaltney Jr., Richardson 2005: 55, 81, Highsmith 2000: 130).

3.3 Problem Definition

The problem definition section is divided into four different categories. All of these are concerned with problem definition but are dealt with separately for clarity.

Define early on

Defining the problem will tell what the software should do. It will give the whole team a broader view of the goal of the project. A clear goal will allow for everyone to focus on reaching it instead of focusing on something unnecessary. (Bjørnvig, Coplien 2010: 68 - 69).

Defining the problem clearly in the beginning and being clear about the requirements will make sure that additional features are not forced in to the software later on. (Gwaltney Jr., Richardson 2005: 159)

The project can never be finished if the problem is unknown. The problem can also be a desired place or a feature in software as well. The important thing is that everyone understands the goal of the project. The problem definition should be revised every now and then, to make sure the project is heading in the right direction. (Condon 2002:21, Bjørnvig, Coplien 2010: 29).

A clear problem definition will give everyone a common goal to strive towards (Condon 2002: 147).

Everyone together

Problem definition is a good place to get everyone together and it gives a possibility to get to know the customers and/or end users better. The project's goal is usually to build something for the customer. The better the customer is known, the better that said customer can be served. (Highsmith 2000: 66-67).

There are a lot of ways to make a problem definition. The best ones however tend to be short, written down, simple and shared. A short problem definition is easy to understand and relate to. Once it is written down, it must be shared. Simple is usually the best with everything and it is also the goal in problem definition. A shared problem definition will engage everyone. The biggest benefit of problem definition is the process itself. Everyone will get together in a team to find the problem. Everyone can say what they think and contribute to the definition which will benefit the entire project. (Bjørnvig, Coplien 2010: 70).

The project can run into problems even if the requirements are specified properly. The requirements can be specified in a vague way or incorrectly. The requirements can also be misinterpreted by the persons responsible for the building of the software. These are problems that software projects are frequently faced with. The best way to fight and prevent these from happening is to incorporate everyone in the project from the

beginning to the end. This way the problem definition and the requirements will most likely mirror what the customer wants from the software. (Krishnamurthy, Saran 2008: 62).

The main objective of problem definition is not necessarily a perfect definition of the problem. It is rather the fact that the whole team is gathered in a single place at the same time to discuss the problem. People will be able to share their opinions, contribute to the project and to really start working as a team. (Bjørnvig, Coplien 2010: 73).

Review regularly

The problem should be defined early on in the project but it should be revived regularly. It might change over the course of the project and in these cases it should be noted as soon as possible. This way everyone can refocus on the new problem and make the necessary changes. (Bjørnvig, Coplien 2010: 68 - 69).

Reviewing the problem definition will also inform when the project is finished. Software manufacturing companies have problems with actually finishing a project. A clear problem definition will remedy this problem. (Gwaltney Jr., Richardson 2005: 160).

Requirements

From a specified problem definition it is possible to start defining the requirements. Requirements are given by the customer and they will specify the problem. As stated before, problem definition should be short and it should clarify the goal of the project. The requirements broaden the scope of problem definition. It is important to have all the stakeholders come together early on to define these things. It is not uncommon for software projects to run into problems because the customer is not getting what they want. This can be because the requirements were too vague or there were some problems with the communication. Defining the problem and the requirements early on with all the stakeholders can reduce this risk or even remove it. (Condon 2002:75, Krishnamurthy, Saran 2008: 59 - 60).

3.4 System Architecture

This subsection is divided into four parts to clarify the subject of system architecture. This subsection also takes a quick look at system functionality. This is to make sure the difference between the two terms is known. The next subsection will focus more on functionality.

Architecture vs. functionality

The architecture basically describes the attributes or qualities of the whole system, while functionality can be seen as describing a defined unit or part of the system. (Krishnamurthy, Saran 2008: 85 - 86). What the system is can be defined as form, or architecture, and what it does as functionality. Both of them should be focused on from the very beginning of the project. The form gives the project a foundation and the main structure whereas functionality is what the end-users specifically need. (Bjørnvig, Coplien 2010: 30).

The architecture of the system means what the system is. The functionality of the system means what it does. Table 2 below shows the main differences between these two. (Bjørnvig, Coplien 2010: 28).

What the system *does*	What the system *is*
User *doing*	User *thinking*
Roles, identifiers	Classes and Objects
End users	Domain experts
User experience people	Architects
Interface designers	Long-term stable structure
Ever-changing functionality	

Table 2. System functionality and architecture (Bjørnvig, Coplien 2010: 28).

Table 2 shows the differences between system functionality and system architecture. What the system is, i.e. the architecture, is concerned with the long-term stable structure of the software. To determine the architecture it is imperative to focus on user thinking. The knowledge of user thinking can be then transmuted into classes and objects that make up the architecture. (Bjørnvig, Coplien 2010: 31).

System functionality, i.e. what the system does, focuses on the changing functionality. It focuses more on what the users are actually doing. The focus is on the features that the end users are going to need. (Bjørnvig, Coplien 2010: 31).

Foundation

The architecture and the affiliated documents give the project structure and a reference point. The architecture usually changes as the project evolves. This is good as it will enable the project to benefit the customer better. (Krishnamurthy, Saran 2008: 88).

System architecture is constructed with code. The important thing with system architecture is that it stays almost intact with similar software projects. That is not to say that it cannot be touched or changed but that there are more similarities than not. Any company can greatly reduce rework by focusing on building an architecture that is solid and usable.

The architecture should be based on the stakeholders' knowledge of the end-users. That will allow for a more critical view of the actual compatibility of the software to the actual users. The foundation is built to be able to stand firmly and also to be able to react to changes. (Bjørnvig, Coplien 2010: 31).

The architecture in lean method is used to construct a tried and tested way of building software. It is used to make sure that small mistakes from repetitive tasks are not found in the software. The architecture reflects the experience of the team members and by providing a stable base it reduces unnecessary rework. (Bjørnvig, Coplien 2010: 83 - 84).

Software architecture is used to give a more simple view of the complexities that software applications include. Architecture gives the stakeholders a better and simpler view of the applications. It also helps to define the interfaces between different components and what those components should accomplish. (Krishnamurthy, Saran 2008: 88).

Ability to react to changes

The benefits of building a good architecture for software are usually not visible at the beginning of a project. A good architecture will bring the biggest benefits when there is a need for a change or a modification in the functionality. (Krishnamurthy, Saran 2008: 85 - 86).

It is usually impossible to say what the end product will look like at the beginning of the project. It is however possible to plan it fairly accurately while still leaving room for changes later on. (Krishnamurthy, Saran 2008: 88).

Enable functionality

Architecture is built to enable the functionality to work properly according to the end-user needs. The end-users or customers usually do not care about the architecture, they care about the software functioning as it should. The objective of software manufacturing should be to make software that works the way it is expected to, not to focus on just building beautiful architecture. A good architecture will make it possible to code good functionality later on and it will help to cope with changes along the way. Building a solid foundation with lean methods will enable faster reaction times to changing environments. With lean architecture in place it is easier and faster to do incremental changes. The form is made up front and after the case studies, the functionality can be built. This way the whole system does not have to be made from scratch, the form is already there. (Bjørnvig, Coplien 2010: 32, 38).

Functionality should however always be the top priority. Working functionality is what the customers are after and it is the most important thing to deliver. (Bjørnvig, Coplien 2010: 81).

3.5 System Functionality

This subsection is divided into two parts that look at the functionality of the system. The previous subsection explained the difference between architecture and functionality. It also provided a quick look at the functionality of the system and this subsection aims at providing a deeper view to it.

End-users

What the system does, i.e. functionality, is going to be services for end-users. Once the architecture is set and stable, the functionality can be defined more deliberately to fit the needs of the process. The end-user roles and the interactions between these roles must be determined as well as the mental model considering these processes. This can be done by use cases. The goal is to find out as precisely as possible how the end-users think the roles should work and how they think these should interact. This will enable to define the mutual dependencies between the roles and their interactions and to see the process from their point of view. (Bjørnvig, Coplien 2010: 32).

What the system does

System functionality describes what the system does. Problem definition gave the main objective that the software should reach. System architecture gave a view of how the software will be constructed. System functionality takes this further and takes on the different aspects that need to be addressed to reach that goal. System functionality can be seen as the features of the software. A feature in software means an improvement to the software. A feature is something that adds functionality to the software. (Gwaltney Jr., Richardson 2005: 40)

It is important to carefully understand what the software should accomplish. A good way to do this is to look at three basic questions and the answers to them. In all simplicity, the questions are Who, What and Why? Who is going to be the end-user?

What does that person want to do with this software? Why would that person use this software? The answers to these questions should give a precise understanding as to what the functionality should be. It's worth underlining that the software is built for the end-users. If they feel it is what they want and need, the software has a good chance to succeed. (Bjørnvig, Coplien 2010: 166).

3.6 Problems with Lean Production

There are some practices in lean methods that are hard to implement when the degree of mass customization increases (Badurdeen, Stump 2012: 110).The main caveats that come with lean are not focusing on the customer as much as one should and not doing enough work because of being lean. It is of utmost importance to focus on the customer as well as everyone involved from the beginning to the end. It does take some extra work but it is not one of those things you can strip off in order to be lean. It is also closely related to the second problem. By focusing on the most important things in a project, what is usually left out, are the things people do not like doing. These include confrontations with customers. Lean cannot be used as an excuse for not doing something necessary. It should be used to rationalize as to why skip something unnecessary. (Bjørnvig, Coplien 2010: 14).

4 Common Failures

Even though the software manufacturing field is very broad, there are a lot of commonalities between projects that overrun their budgets or deadlines or that do not deliver what they promised. This section focuses on the most common failures found in the software manufacturing field and how to avoid them.

The earlier sections have looked into ways to make software manufacturing as efficient and productive as possible. Theories however often run into problems when it is time to put them into action. Real life situations vary greatly from what we think should happen. By looking at the mistakes and problems that other companies have encountered, it is possible to try and avoid these mistakes. The mistakes discussed below are definitely not the only ones a project can encounter but a few selected of the most common ones. (Gwaltney Jr., Richardson 2005: 129).

4.1 Oversimplification

Simplification

It has been suggested that the flawed human thinking process is the reason systems failure. The human mind does not process different kinds of information with rapid pace. For example, humans have a hard time multiplying large numbers. This innate challenge forces humans to simplify systems into more understandable and therefore simple versions. (Krishnamurthy, Saran 2008: 9).

Making software manufacturing processes simple makes sense. There is no reason to make things more complicated than they need to be. Einstein is credited to saying *everything should be made as simple as possible, but not one bit simpler*. The latter part of the sentence is what stems problems. People like to look for certain patterns and structure in everything. The problem is that in software one cannot always find it. It does not mean that the software is not structured or built properly. It just means that one cannot decide on a certain structure and then try to fit the pieces of software into it. For example sometimes it is necessary to build two different functionalities even though it would be desirable to fit everything into one. (Highsmith 2000: 182).

Oversimplification

A difficult aspect of oversimplification is how to decide whether something actually is oversimplification or not. Just focusing on not making things too simple will probably do more harm than not. A good architecture can greatly reduce this risk. Getting a view of the overall project and the parts of it is a good place to start. When specifying for example what features are needed, it is easier to see if they can be situated in the existing objects. This way the project can be kept as simple as possible. If there is need to add a new object to the architecture, it can be done with a certainty that it is necessary.

Simplifying is not a problem on its own but can easily lead into oversimplification. Oversimplification can result in focusing on just a few core objects or processes and discarding the complex interfaces that actually make up the software. This can lead to a false belief that one is in control of the process and carry on the process. It is also a problem that people do not like to give away their professional self-esteem. It can be really hard to admit not completely knowing or understanding the relations of processes in a project. But oversimplification can lead to decisions that do not necessary support the goal of the software. This might not even be a problem or it can mean a complete failure of the software later on. (Krishnamurthy, Saran 2008: 9).

Prevention

Oversimplification is a problem that can easily be solved. What is most important in preventing it is the realization of what it is. It is not something that anyone does on purpose. It happens when a process is being optimized and it gets taken too far.

To prevent these kinds of problems, one should always document which assumptions have been made, at which stage and which processes do they affect. Simplification is in no way a bad thing. It is almost a necessity in making a software manufacturing process efficient and agile. It is however important that at any point one can go back and see which simplifications has been made. This will not only help in preventing sys-

tem failures but also in fixing problems that might come up. (Krishnamurthy, Saran 2008: 9).

4.2 Poor Communication

Communication errors are a major cause of failures in software business. The underlying issue in software field is that a lot of times companies have separate sales team and programming team. While this is completely understandable in terms of different knowledge base needed, it can lead to specification and costs estimate problems. It might be hard for a programmer to say with a complete certainty whether or not two different functions of software can operate simultaneously, let alone for a sales person. A sales person might also not realize into how deep the specifications should go for a required feature. Leaving requirements open will leave the programmer guessing or having to contact the customer on a regular basis, costing time and patience. (Krishnamurthy, Saran 2008: 10 - 11).

Poor communication is the most difficult and most common of these problems. First of all, it might be hard to recognize. For example, it is a lot harder to state an opinion clearly, unless the other person understands the subject. This can become a problem when a designer and a programmer are interacting and trying to understand a problem.

Face-to-face communication is far superior to any other type of communication and cannot be replaced with emails or phones. Face-to-face communication allows people to work more like a team. (Gwaltney Jr., Richardson 2005: 147)

A communication problem is often an emotional issue since the problem is not usually just a single person. It can be even project wide. This can be because of persons involved or because of communication methods. It might also be hard to try to convince people to start improving communication in a project when there is a deadline approaching. It would be beneficial to divide the communication problem into two separate issues, one concerning the overall communication methods and the other concerning a particular project. This is mostly to ease the improving of this process. If the problem is just the particular project, it is most likely because of the personnel, not the

communication methods. Therefore it would not be wise to alter the existing methods; rather one should focus on the people. The bottom line is to make sure whether the communication problem, if one exists, is caused by a person or if it caused by faulty methods.

The working environment will be much more collaborative when everyone knows what others are working on (Condon 2002: 142).

Communication problems are hard to improve and even harder to measure. When can a company state with certainty that the communication in now working? Communication should be thought of as something that needs to be improved continuously. Communication can never be too fluent but it can often be in need of improvements.

Real life projects also face a lot of changes. Someone might change a job or get sick for example. The negative effects these situations have can be mitigated when others know what is going on in the project and work can be redistributed.

Guessing is even worse since it can lead to unwanted features and cascading failures later on. Estimates on the costs of the project can also change with even a seemingly small new requirement. It is of paramount importance that a person who will actually be a part of the programming team, or at least is a skilled programmer, sits in the meeting and participates. It will decrease the possibility of underestimating costs by the sales people. This way the company will also not make any unrealistic promises as to what the software can deliver. (Krishnamurthy, Saran 2008: 10 - 11).

4.3 Unhappy Customers

Happy customers are great marketing. Keeping the customers happy should be number one priority in any project. Happy customers will want to do business again and they might even bring in new customers. The software is made to them and it should reflect what they want. The best way to make this happen is to make sure the customer is in on the project constantly. Customers should be kept informed about the progress and they should have a possibility to tell what they think. If the project is not

going in the right direction it should be steered another way. (Gwaltney Jr., Richardson 2005: 129).

Keeping the existing customers happy is the best way to ensure a strong customer base. Customers who are not happy with the service or the products will most likely move their business to another company. Manufacturing products that are of high quality will enhance the company image as well as improve customer satisfaction. (Pfeifer, Schmitt 2009: 120).

In addition to good quality products, customer satisfaction is heavily influenced by emotional bonds. Customers should always feel that the company delivering the product actually cares about them. The customer needs to be listened to and involved in the process all the time. (Beelaerts van Blokland, Curran, Verhagen 2009: 836).

Keeping customers in on the process from the beginning to the end will improve customer satisfaction. The customer can inform whether the project is going in the right direction or if something should be altered. (Gwaltney Jr., Richardson 2005: 140)

4.4 Sidetracked Workers

There should be regular meetings and collaboration between the team throughout the project to keep everyone on track. It is not unusual for employees to get sidetracked when working on a project. A meeting where everyone informs what they are working on and what they have accomplished so far will make it easier to stay on track. When a person has to explain what they have done and why, they might realize themselves where they have gone wrong. It is also the responsibility of other workers to make sure everyone is working towards the same goal. Oftentimes people become blind to their own work and this is a time when co-workers have to step in. It might be hard to tell a person that a day's work has been lost but it is a lot better than losing a week's work or not producing the precise functionality that is required. (Gwaltney Jr., Richardson 2005: 142).

These are hard situations since the person getting sidetracked is usually not doing it on purpose. The person might be doing extra work in order to help the project and to

contribute. Sometimes all that is needed to make these kinds of problems go away is to have these meetings. Frequent meetings help keep everyone on track. (Gwaltney Jr., Richardson 2005: 80-81).

People will realize that they are not doing what is necessary or right and will switch their course. Sometimes however more is needed. The manager could have some one-on-one time with this person and go through what is needed and what is not. It is not a good strategy to start blaming or accusing people in front of others in a meeting. Good feedback should be given in public, bad feedback should be given in private. (Gwaltney Jr., Richardson. 2005: 143).

4.5 Absence of Emergence

An important property of a successful software team is emergence. Emergence stems from the interactions of team members and creates a behavior for the whole group. It can be thought of as somewhat similar to innovation or creativity. The problem with emergence is that it is not easy to explain or understand. Everyone knows what it means to be in a 'flow'- state. It is a whole different thing to try to understand how to get there and why it happens. Emergence allows teams to build their own behavioral patterns that will lead to successful results. (Highsmith III 2000: 31).

People will be able to bond better if they can speak freely. Informal meetings allow everyone to speak without putting anyone on the spot. This will foster emergence in the project. (Gwaltney Jr., Richardson 2005: 81)

A good way to destroy emergence in a complex project is to impose strict rules to everything. Emergence strives from the project team's possibility to react to situations, to be adaptive, to do things in their own way. Imposing a lot of rules for a project can work in a simple project but it is definitely not the best way to work on a complex project. Simple rules and the ability to have good relationships with other team members is what will enable emergence. (Highsmith III 2000: 31).

4.6 Lean Methods and Common Problems

Figure 4 below illustrates the connection between lean methods and common problems. On top of the figure are the most common problems found in software manufacturing and below are the lean methods than can be used to deal with these problems.

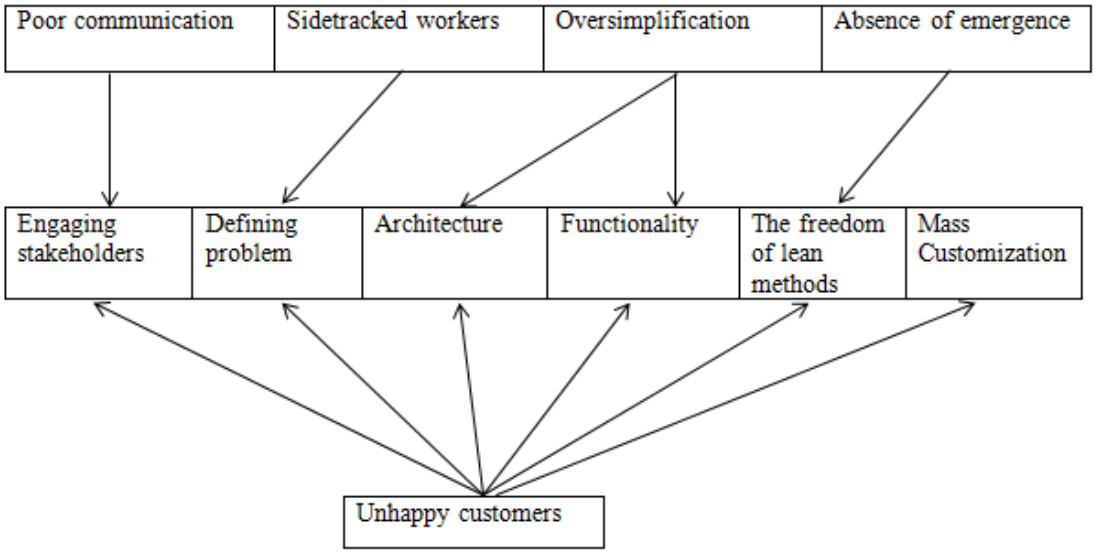

Figure 4. Connection between common problems and lean methods.

Figure 4 above is a simplified version of the connections between the methods described in this thesis and the problems. The arrows are used to show which method can be used to fix a particular problem in a company. The freedom of lean methods refers to the idea that in lean there are no strict rules of conducting a project. Lean methods focus on being a framework, not a set of strict rules. The goal is to get to the particular goal at that time, but the ways of working may vary.

Unhappy customers are the hardest problem to solve since it is affected by all the methods. Therefore a company with satisfied customers is very likely a successful one. The goal of all software manufacturing should be to deliver what the customer wants and make them satisfied.

It would be nice to think that using these particular methods, the company could eliminate these particular problems. While it is true to some extent, there are a lot more factors that play into these problems. The methods, problems and links between them

do however exist and using these methods can greatly reduce the risks of these problems. They are just not the end all be all for these problems.

5 Summary of Theory

This section summarizes the theories used in the previous chapters.

The theories used in the previous sections have looked at software manufacturing from different points of view. The first part looked at how the software manufacturing is arranged in mass customization. The next part examined at the manufacturing processes from the lean methods point of view. The last part took another view and looked at the most common problems found in software manufacturing and how these problems could be avoided.

The goal is to compare the findings from the theory with the information from Hammerkit. Table 3 was made to help understand the difference between the findings and the status of the processes in Hammerkit. The table includes the parts that the author thought are the most important ones according to theory and the goals of Hammerkit. The table will be re-introduced after the next chapter to illustrate the findings from Hammerkit. Since there are overlapping's in the theories, as is always the case in real life, the questions are not sorted under headings. Sorting them under headings would be a problem when the questions concern more than one section of the theory.

The questions were made in a manner that answering could be done with *no, almost, yes.* This can sway some of the questions into being leading. The focus was however not to have leading questions. Open questions would however not have had that much value for this thesis, since they were answered by the author.

The questions were formatted with the focus on the whole theory. The goal was to get an understanding as to how lean practices work in manufacturing. The mass customization was given thought as well as the common problems. The questions try to answer for the whole theory and give a good understanding as to where Hammerkit is.

#	Statement	No	Sometimes/ almost	Yes
1	The processes and methods concerned directly with the end user			
2	The products customized for a particular customer			
3	There is a large product offering			
4	The main focus is on people, instead of methods			
5	The processes produce value all the time			
6	There is a framework from which to work			
7	There is a continuous process flow			
8	Everyone is in on the project from the very beginning			
9	Most of the communication is done face-to-face			
10	The goal is defined early on			
11	There are regular informal meetings			
12	There is a focus on customer expectations			
13	There is a focus on eliminating waste			
14	The goal is shared by all the team members			
15	The architecture gives a top level view			
16	The customers are kept informed about the progress			
17	There is a way to recognize/measure if the communication is working			
18	The employees are aware of what others are working on			

Table 3. Questions for comparison.

There is some overlapping in the questions, as can be seen in table 3. This was to be expected as the goal was to have a thorough base for making the comparison with Hammerkit.

6 Current Status and Findings

The section on current status and findings takes a look at Hammerkit and describes where the company is now and what the main findings are.

6.1 Stakeholders

The stakeholders in this subsection are considered to be everyone connected to the project.

The stakeholders from the customer side are in on the project from the very beginning. People from Hammerkit and the customers' side have a meeting to discuss the specifics of the project. The customers explain what they need and Hammerkit explains what they have to offer. The cloud store platform is something that comes with the formats. What the customers are looking for are the formats that are put in the cloud store.

Hammerkit starts the whole process by finding out what the customers want exactly. Public Relations agencies have a lot of similarities regarding their needs and wants. Therefore it is possible to manufacture some examples of formats that Hammerkit could produce for them. These formats are then shown to the customer and there is a discussion concerning what the customers want. The customers inform Hammerkit of what they want in terms of visual appearance and functionality. These needs are written down and the design will begin.

The meetings between the customers and Hammerkit are held face-to-face. From Hammerkit, usually present at the meetings are the programmer or programmers for that particular project and someone from the management. All the stakeholders are kept in on the project from the beginning to the end. Once the problem has been defined and the requirements are known, the manufacturing can start. The customers are kept informed about the development throughout the process. Additional meetings are held as needed, sometimes up to once a week. The project team working on the software at Hammerkit meets daily. They are working on the premises so these meetings are not as formal as with the customers.

Findings

All the stakeholders are taken in on the project from the very beginning. The customers are met face-to-face to get to know them. The people involved in the project will meet the customers to discuss the project. This way the possibility of misunderstandings is also reduced. The customers are kept informed about the progress throughout the project and met with regularly. People can also be in contact via email and phone if needed.

The project team works under the same roof and therefore can communicate as much as needed. These meetings are informal and help to build trust between the team. The team members can also help each other easily when working in the same place. A more collaborative environment can also be achieved when people know what others are working on.

Hammerkit has at least someone from the business side as well as someone from the programming side in the first meetings. This helps the customer as well as Hammerkit understand what the costs, the schedule etc. are going to be. There will not be any unrealistic claims as to what can be delivered and what the costs are. There will also be different points of view.

6.2 Problem Definition

The problem definition is accurate in the case of cloud store formats. When a customer wants for example a format for a Facebook quiz, it is easy to define the problem. The problem and the main goal of a project are defined during the first meeting or meetings. The customers can decide how many formats they want and what kinds of formats. The basic structure of the cloud store will not be changed, only the formats. Customers can also decide how far the manufacturing of the individual formats is taken.

The customer expectations are studied even before meeting with the customers. Hammerkit sets out to find out what these kinds of customers do in their business. They want to know who their end users are and what they are going to do with the software? Figuring out what the software should accomplish and what it should look

like is the beginning of the manufacturing. Based on these findings, Hammerkit designs the formats for the customers. Once the design is ready, the formats are sold to the PR agencies. Then the final stage of the manufacturing is the actual coding of the formats for the offices. The order of the process is very common in mass customization, i.e. design-sell-make.

The customers are given examples of what Hammerkit could produce for them. Customers can then give their opinions. What requirements they have, what should be changed etc. These requirements and demands are written down and the format is produced. The customer is shown the new format and if needed it is fixed again. Once the proper form and outlook is found, the format is sold to the customer. This format is then manufactured to the customer's cloud store.

The basic formats that these public relations agencies use have a lot in common. Agencies also have more specific formats that they use that other agencies do not use. These formats of course need to be defined more accurately. These formats, as the more common ones, are defined face-to-face.

Findings

Problem definition is done in the beginning of the project as it should be. All the stakeholders are together and can communicate freely face-to-face. This way the end product will most likely be what the customer expected. The requirements are decided together. There will not be any requirements that the customer does not want. The requirements will also be possible to manufacture.

Problem definition can be reviewed regularly, inside Hammerkit as well as with the customer, since problem definition is in written form. This way the project will flow in the right direction. It will also help employees to not derail away from the end goal.

Hammerkit uses weekly meetings in order to keep everyone informed about what others are working on. These meetings are short and the whole purpose is not to waste too much time on a process that does not create value on its own. Workers inform each other about what they are working on, what they have done and what they are about to do. Others can ask for more information after this if they need.

This is a process that works well if the company is small as Hammerkit. There is no need to get everyone to a separate room or produce unnecessary documents for the sake of producing them. The focus should be on producing the bare minimum needed. A great benefit is also that others can question someone else's work. There needs to be a reason to do something. This way employees will not get sidetracked for more than a couple of days at a time. There are more specific meetings in projects. People involved in manufacturing software will meet to discuss the details of the process.

6.3 System Architecture

Hammerkit lets their customers know what they can change in a format and what should be left untouched. For example, they can change the company name on the page, but should not modify the code that states where it is located to avoid breaking down the layout of the page. In this context we can think of the architecture as the part that should be left untouched. It gives the software the structure that has been defined early on and thus there should be no reason to tamper with it.

HammerKit, the tool that is used at Hammerkit, is a visual development environment and therefore a mind map is an appropriate way to visualize it. While the architecture of any site is basically code, this picture gives a better view of what the architecture behind a web site is.

Here we will use a *Software Manufacturing Standard* by Mark Sorsa-Leslie from Hammerkit to review how the architecture is managed nowadays. The picture below shows how the different parts of a web page relate to each other.

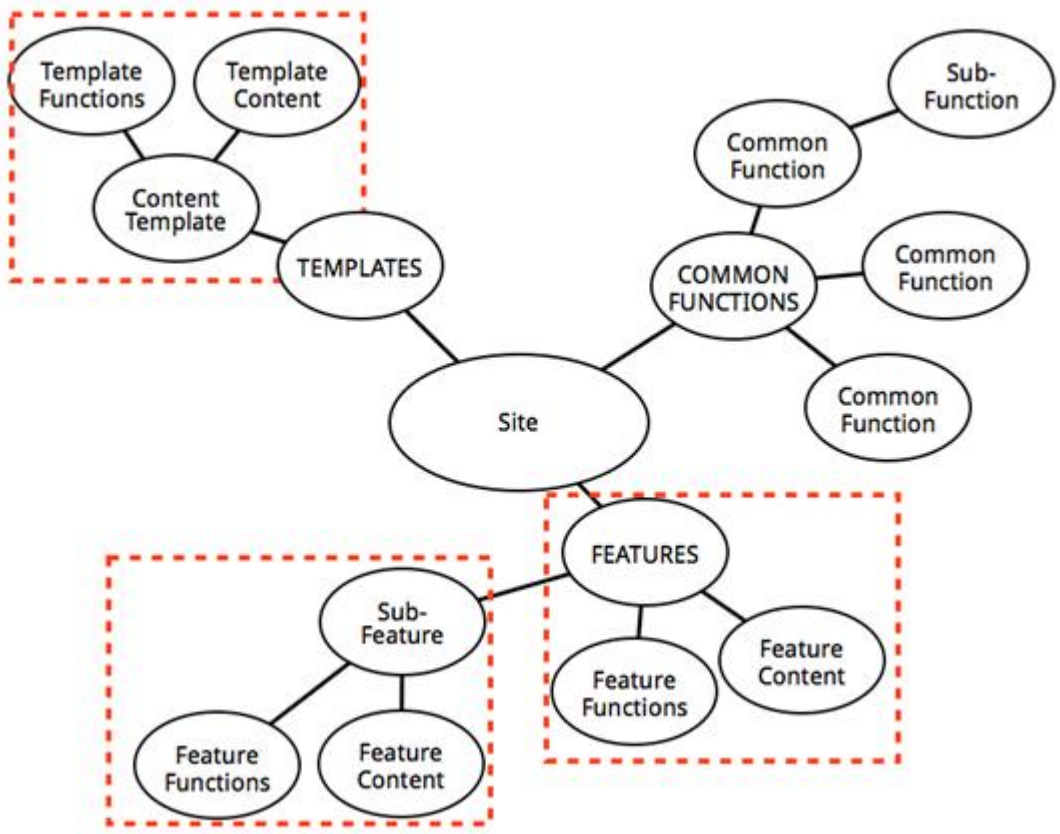

Figure 5. Standard outline structure of a site.

A short description of the terms in figure 5 is in order to help understand them. Site is the collection of different web pages for that project. Common functions refer to the things that most of the pages in this particular project have. Templates are basically the foundations of the pages. The functionalities are built on top of these templates in order to have a working page. Features are things that these pages can do or show, for example a shopping cart feature on the site. A subfeature would be an extension to a feature, like the possibility to delete items from the shopping cart.

Now from here on it is easier to determine which of these should be considered as architecture and which should be seen as functionality. It is however not an easy task. It would be beneficial to have clear borders but real life software manufacturing does not work that way. The simplest and probably best way would be to determine that the first steps represent the architecture and the next steps should be considered functionality. This means that the architecture would consist of *site, templates, common functions* and *features.*

Findings

There is a unified way to structure the sites. This holds true from project to project as well as for each and every programmer. It makes it easier to detect any bugs in the software. The sites have a clear foundation and structure. The customers are taken in on the project from the beginning and that way the architecture can be based on the knowledge of the end-users.

The way Hammerkit is structuring architecture, as shown in figure 5, is a good way to structure the architecture. It is a simple top level view of what a site should contain. There are a few tangible benefits from having architecture like this. Firstly it is for everyone to see. Having a base to work with that is the same for everyone will help with the rest of the project. Everyone in the project knows how it should be constituted and will work to make it happen. It is also easier for team members to help each other since they know the structure of the software.

The architecture is simple and allows anyone to easily see how all the different parts relate to each other. This allows programmers to react to possible changes rather quickly. A simple and working architecture also makes it possible to add the important functionality to the site.

The architecture is built in a way where each new common function and template has its own place. This should be enough to greatly reduce the risk of simplifying things too much. It is easy to decide whether to build a new object to the site or not. If the function is not built before, then a new object needs to be added. It keeps things simple enough without going too far.

6.4 System Functionality

As with architecture the functionality is also very dependent on the end-users. In order to build the desired functionality for the customers, Hammerkit finds out who is going to be using this software. There will also have to be accurate knowledge as to what

the end-users are going to be doing and why they are doing it. These questions will be answered by the customers or the end-users, depending on the software.

The functionality part will give the customers additional features and therefore all the information that was gathered from the customers and end-users will be beneficial. The next thing is to decide how to arrange the functionality within the software. The idea is to keep it simple, effective and reliable.

Figure 6 shows the outline of a site and how the functionality is arranged.

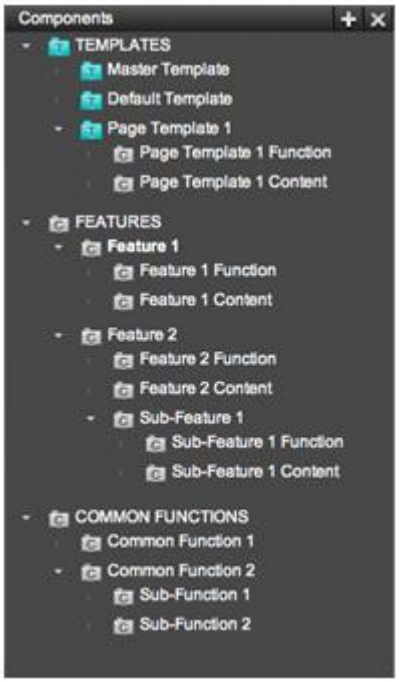

Figure 6. Outline structure of HammerKit.

Hammerkit has taken an approach to software manufacturing that can be seen in the picture above. This represents the same objects as the picture used earlier in the chapter on architecture but this time it is pictured in the HammerKit development environment. This is a representation of a *site.* From the previous chapter one can also see the *templates, features* and *common functions*. As noted in the theoretical part, architecture and functionality are not two completely different issues. They are interlinked to some degree.

The functionality on this page would be found in the feature categories. One might also argue that the template, or how everything is arranged on the site, is also part of the functionality. To some extent it is but in this case it will not be considered as functionality but as a part of the architecture.

Figure 7 shows a web page done with HammerKit to give a better understanding into what functionality can be. On the upper right there is a text that says *Ostoskori (0,00 €).* This is a feature and is considered a part of functionality. It is a shopping cart and when shopping online, it is used to show the amount of money the items will cost.

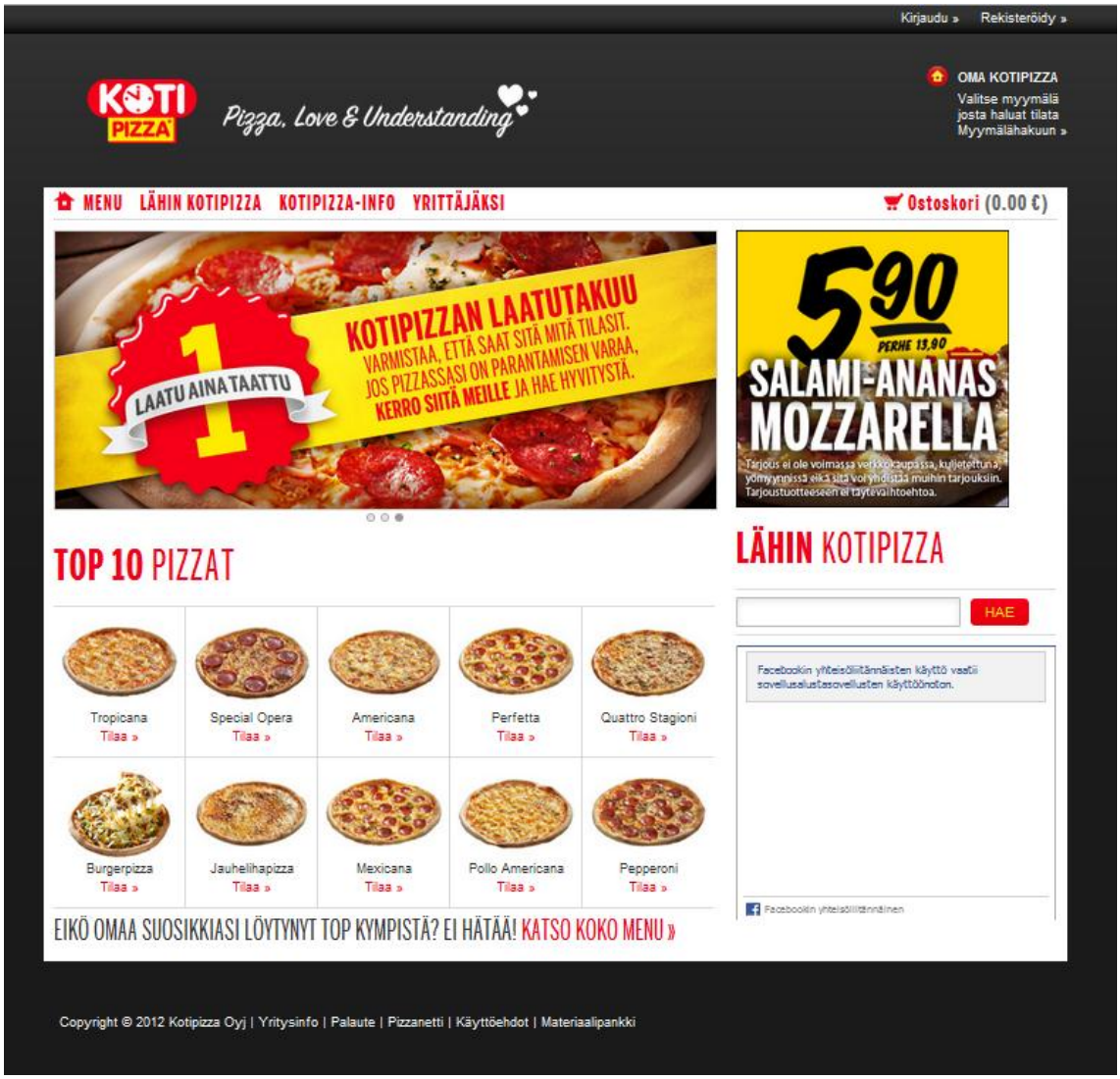

Figure 7. Koti Pizza's home page. http://www.kotipizza.fi

Functionality is something that the customers should also leave untouched. Touching the code in functionality might impair how the format works.

Findings

Hammerkit focuses hard on understanding what the customer wants. Once all the necessary information is gathered the additional features will be built. The features are rather easy to arrange since the architecture was built simple enough. The functionality is given a lot of focus. This is good as it will be the most important thing from the customer point of view.

The different functionalities are spread into individual objects in Hammerkit. This will allow the programmers to focus solely on the functionality at hand. It is easier to see how far the project has gone and what still needs to be done when the functionalities are handled as individual parts of the whole system. It is also beneficial that it is easy to see which subfeatures relate to which feature. This way it can also be quite easy to trace if a part of the software is not working. Also by constraining the features into individual objects one can fix them easier without affecting the rest of the software.

There is however also one drawback to putting features into individual objects. The team can lose sight as to how each of these features works with each other and the system as a whole. It is not a major problem but rather something that needs to be kept in mind.

Functionality is very similar to architecture when looking at simplifying things. It is clearly structured and there should not be a great risk of oversimplifying things. The visual build in HammerKit makes it easier to see whether things have been simplified too much or not.

6.5 Mass Customization

Hammerkit is currently in the process of implementing mass customization. It is used to achieve a broader customer base. The whole process is started by contacting public

relations agencies. These agencies have numerous offices that mostly use similar tools to achieve their goals. These tools can be campaign sites, radio services, social intranets etc. The particular demand for different sites is determined from the agency itself. This way there will be no guess work as to what the customers want or need. It will also help to determine how far mass customization can be taken. Which are the parts that will be left untouched by the customer and which parts will be modified.

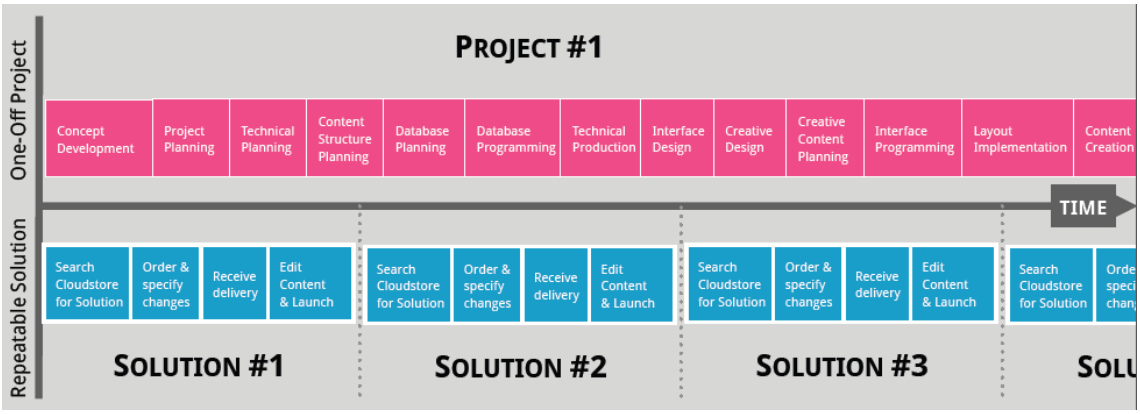

Figure 8. Difference between one-off project and repeatable solution.

Figure 8 above illustrates the difference between a project that is done from scratch to the end and a solution that can be used again and again. Building repeatable solutions like this is what Hammerkit is focusing on. While a simplified figure, it still pictures clearly how mass customization can be of huge benefit. The product life cycles and development cycles can be shortened.

Once the demand is in the offing, the actual manufacturing can start. The customization was taken into consideration when determining what the particular customer wants. Switching to mass customization requires the software to be produced to the masses. This will be done by building core software. This can be for example the aforementioned social intranet. This format is manufactured as far as possible while still preserving the individualism needed. Individualism can mean a variety of things. It can be the color of the format, company logo, the content or something else. The important thing is to develop the core software far enough, so that the customer will benefit from not manufacturing it itself, while still leaving enough room for their own content.

Mass customization is achieved by manufacturing solutions that can be repeated. All the offices can use the same solution format and add their own content. This way a particular format is produced only once but it can be used multiple times. Below are some examples of repeatable solutions.

Corporate Comms	Social Media	Mobile
Crisis Management System	**Facebook Competition**	OneSite
Coverage Monitoring	**Social Aggregator**	Mobile Newsletter
Social Intranet	**Blogs**	Mobile Commerce
Dark Sites	Google+ Plus integrator	Mobile Support
Digital Newsletter	**Campaign community**	Booking/Sales
Digital Annual Reports	RT Social monitoring	**Corporate Repository**
Corporate site	Polls and analytics	Mobile Blogging
Event/Multi-Event Site	Facebook Recruiting App	Mobile Video Production
Corporate Video Platform	Corporate Microblogging	DOOH Campaigns
Virtual Data Room	**Facebook Quizzes**	
Media Bank	**Facebook Commerce**	

Table 4. Examples of repeatable solutions in the cloud store

Table 4 illustrates different kinds of repeatable solutions that Hammerkit is offering to their customers inside the cloud store. A company decides that they want some of these formats and Hammerkit builds them for their cloud store. An agency probably would not want all of these formats but rather pick the ones that fit their particular needs at that point. These formats can then be bought from their corporate application store.

Adding uniqueness to software is what the customers will do. It is what separates them from their competitors. What Hammerkit does, is provide the format on which that uniqueness can be added. This is done with the repeatable solutions. Making good

software and then producing it in mass quantities. This way the customer can focus on what they do best and not having to worry about all the programming that goes into producing the formats.

Findings

Hammerkit makes sure they know what the customers actually want. These formats are then designed and made to the cloud store. Customers can then buy the formats from the company's cloud store. The customization is made up-front by figuring out what should be produced. The mass phase is achieved with the cloud store. Once the format is ready, all the different offices can purchase it from the agency's store. The format is only made once and it can be re-used again and again.

This of course means improved efficiency by reducing the rework needed. It will also allow for a successful format to be used again and again. While all formats are of course made as good as possible, some formats just work better than others. These formats will be highlighted in the cloud store so that all the offices can use them.

Highlighting the most popular formats will benefit the customer by helping them use the most functional formats. Mass customization also reduces the costs for the customer by reducing the production time and costs for Hammerkit. Customization makes sure that the customer gets what they want.

Customer satisfaction will always be a major concern. The whole manufacturing process however is initiated by finding out what the customers want and need. It is a lot easier to deliver what should be delivered when the company has an understanding of the customer expectations.

6.6 Summary of Findings

This subsection will compare the theory and Hammerkits' information more thoroughly. The previous subsection examined the processes and methods at Hammerkit. The findings from that subsection serve as the basis for this comparison. The comparison is made using table 3 presented in section 5 and the results can be seen in table 5.

#	Statement	No	Sometimes/ almost	Yes
1	The processes and methods concerned directly with the end user			X
2	The products customized for a particular customer			X
3	There is a large product offering			X
4	The main focus is on people, instead of methods			X
5	The processes produce value all the time		X	
6	There is a framework from which to work			X
7	There is a continuous process flow			X
8	Everyone is in on the project from the very beginning			X
9	Most of the communication is done face-to-face			X
10	The goal is defined early on			X
11	There are regular informal meetings			X
12	There is a focus on customer expectations			X
13	There is a focus on eliminating waste			X
14	The goal is shared by all the team members			X
15	The architecture gives a top level view			X
16	The customers are kept informed about the progress			X
17	There is a way to recognize/measure if the communication is working		X	
18	The employees are aware of what others are working on			X

Table 5. Results of comparison.

The results were mostly as expected and no major surprises were found. Hammerkit makes their profits from software manufacturing so the processes and methods were expected to be on a good basis.

Based on the comparison there are two things that should be improved. The two main points were:

- The processes do not produce value all the time.
- There is not a way to recognize/measure if communication is working.

There are always ways to improve processes, no matter how well they are working. The main focus should however be given to the things that scored the lowest. The other aspects can be kept on the same level as they are now since they were found to be working well. These processes can be improved later on, when the two others are on the same level. This is why the two aspects that scored the lowest on this comparison are given the main focus.

A company can focus on improving the things that are working well, improve the things that are not working well or do nothing. The best alternative would be to im-

prove the things that do not work as they should. That is why the main focus will be given to these things.

There are also reasons to look at customer satisfaction at Hammerkit. Hammerkit is moving to a new market and it brings new customers. Customer satisfaction and the two other aspects are given a closer look in the section on recommendations.

7 Recommendations

This section will give the author's recommendations for the two aspects found to be in need of improvements. It will also give recommendations as to how to go about improving them, the costs and the benefits for Hammerkit. The aspect of customer satisfaction will also be handled.

7.1 Continuous Value Production

This section focuses on continuous value production in Hammerkit.

Producing value all the time is the goal in lean methods and it should be for any company. The fact is that there will never be a point when value will be produced from the beginning to the end. The goal of thriving towards that point is what is most important. This process is working in Hammerkit and they are working on improving it. The findings indicate, however, that it seems like something that should be focused on even more.

Continuous value flow includes producing value but a huge part of it is also eliminating excess waste. Reducing excess waste will free up time to produce more value for the customer. Manufacturing software is what brings the customers value. Anything that helps or improves this can be considered a method to increase the value production.

Process of improvement

There are numerous ways to improve the amount of value produced for the customer. Hammerkit should figure out where they are now. After finding out where they are now they can start to focus on improvements. Reviewing the processes and the need for improvements should always be done first. This way a process that produces value will not be eliminated. If there however should be a need for improvements then necessary steps to achieve it should be incorporated. The most effective ways are to:

- Decrease the amount of unnecessary documentation.
- Focus on good communication.
- Decrease the amount of strict policies and bureaucracy.
- Manufacture exactly what is needed.

It is no wonder that the list includes things from the theory of this thesis. Decreasing the amount of unnecessary documentation means documents that are not crucial for a project. There should always be documentation about the process, its progress, the assumptions made in the manufacturing of software etc. The focus on good communication will be discussed in detail in the next section. Decreasing the amount of strict policies and bureaucracy will benefit the employees by improving the working environment. Manufacturing exactly what is needed should always be the goal of any company.

Some of these principles are easier to implement than others. What Hammerkit should do is decide whether these things should be improved, and if so, which ones. The best way is to start with item on the above list. There is no need to do everything at once. Hammerkit should take a few projects and review them to see where they are. One project is not enough to inform about the need for improvements since projects differ from one another. After the review, pick one thing and focus on improving it. These are not things that can be done overnight, but rather incrementally.

The list above is only a portion of the vast amount of variables that contribute to value production. Which is the biggest obstacle in Hammerkit is hard to say. That is why it is important to identify the possible aspects that could be improved. Improving the main obstacle in a value chain will produce the biggest benefits.

Benefits

These things focus on reducing the amount of waste, time used not producing anything of value, as well as reducing re-work. The benefits from continuous value production are multifaceted. It reduces the life cycles of products as the amount of re-work will be decreased. Light bureaucracy will also make it faster to produce software. Producing the products faster will allow for cheaper prices which can lead to increased customer satisfaction.

Manufacturing exactly what is needed will shorten the time it takes to finish a project. This can be achieved by making problem definition in the beginning and specifying clearly what the goal is. This way the project will flow into the right direction and the

need for re-work can be reduced or removed. It will save time and therefore produce more value over time.

The goal is to not produce documentation for sake of documentation. There should always be legitimate reason for documenting. This way no extra time will be wasted in documentation that is not read by anyone. The information in the documents are also easier to find when there are less information overall.

Decreasing the amount of strict policies and bureaucracy will increase the value produced by affecting the way employees work. A framework from which to work is a lot better in encouraging emergence in employees. Heavy bureaucracy will lengthen the time it takes to finish a project. A lighter bureaucracy will allow people to work more freely. Strict policies will make it a lot harder to be innovative. People who have the possibility to work freely will be more motivated and more productive.

Costs

The costs in improving these processes come from the time spent in improving them. Optimizing processes and removing unnecessary steps does not cost anything and will be beneficial in the long run. Even though it takes time and effort to improve the processes in the beginning, it will pay off in customer satisfaction. These improvements do not require any new hardware purchases and might not even require new software. It does take some investment from the employees.

7.2 Communication

The question was about measuring or recognizing if communication is working. The underlying problem is most likely to be however the fact that communication and communication methods have not been given a lot of thought. This is understandable, since focusing on communication does not directly influence anything and is really hard to recognize.

The communication problem is similar to the other two problems, as it is affected by a host of things. The biggest thing noticed from Hammerkit was the absence of documentation or information concerning communication. This might be because a small

company can easily work without written documents concerning the ways of communication. There might be problems down the road when the company gets bigger and employs more people. Communication with customers can also face problems. Communication in this context will be thought of as company-wide, not personal. Therefore improving the communication should be done via common methods and/or agreed upon ways of communication.

Communication affects everything a company does. Therefore it should always be a top priority. Getting to know where Hammerkit is now and slowly implementing methods for better communication would be ideal. Even on a company-wide level, communication is still always a personal issue. It is never wise to start rushing issues that people feel are personal.

Process of improvement

Improving communication in Hammerkit should began by figuring out the starting line, where the company at the moment. This can be done via discussions, interviews and/or an outside consultant for example. Once the main aspects in need of improvement are identified the improvement process can begin.

The aspects that are going to be improved affect how the end result will turn out. Companies also use different kinds of solutions to arrange their communication. The best solution would be to produce a document for the whole company. The document should be constructed more like a framework than a strict set of rules, as is also the goal in lean methods.

This framework should set rules for company-wide communication. It should give general guidelines as to how to handle communication in Hammerkit and/or with customers. Communication concerning informal meetings should not be tampered with rules. The biggest benefit of these meetings is the fact that people can freely communicate and interact. Rules would destroy this free atmosphere.

The main focus in this document should be given to communication with customers, communication in projects and communication between employees. All of these situa-

tions have different challenges and therefore should have different frameworks. Things to document should be:

- Which situation is it?
- Which channels of communication to use?
- How often to communicate?
- Who will be in charge of the communication?
- Who will be in on the communication?

The situation means whether it is a business meeting with a possible client or Hammerkit's internal communication. Communication channels is used to describe whether the communication will take place for example face-to-face or through email. There should be a general guideline as to how often to communicate on average in a specific situation. It also needs to be known who is in charge of the communication. The people involved in the communication need to be specified so everyone who needs to be informed will be.

Communication should be monitored once the improvements have been made. This way it is possible to know whether the improvement has benefited Hammerkit or not. Earlier section discussed the importance of reducing waste and unnecessary documentation. It should be noted that communication documents are not unnecessary documents.

It is hard to give specific instructions on how to improve communication as every situation is different. These principles are however universal and can be incorporated in Hammerkit. The main point is to have some structure into everything. It has been stated earlier in this thesis that there should be frameworks, not strict rules. That applies here as well. The point is to get to a point where employees would know how to communicate in a specific situation.

Benefits

Good communication decreases the amount of misunderstandings which decreases time that would have been otherwise wasted. Communication that works will also decrease unnecessary waiting times. People know who is in charge of which communication and who needs to be informed about the progress of the project. This way people who need certain specific information will get it as soon as possible. There will also not

be any guessing as to who will contact whom. This way no employee will be disturbed with issues that do not concern them.

Benefits from good communication are hard to pinpoint since they can be seen on each and every process. There are basically no phases in software manufacturing where people do communicate. A working communication therefore affects almost everything that is done.

Costs

Costs for producing documentation for communication are not huge. It does take some time and employees need to be involved. The process of manufacturing this document can be highly beneficial. Everyone can get together and discuss the communication and how it will work. The document will serve as a reminder once people have learned the ways of communication in Hammerkit. Once the communication issue has been settled, there are no hidden costs. It takes a bit of time, money and effort in the beginning. Then it is done and the benefits can be seen. The benefits that come from improved communication will outweigh the costs it takes to make them.

7.3 Customer Satisfaction

Another important thing to focus on is customer satisfaction. Customer satisfaction should always be a major concern. Hammerkit is also just starting to roll their mass customization into business. This means that the customer base is going to change and Hammerkit needs to build new customer relations.

There are two really important things that need to be taken into consideration. The first thing is to genuinely focus on making the customer happy. This is the only way to survive in a new market. The second one is to keep track and monitor the customer satisfaction. Hammerkit has had really happy customers in the past and some of that reputation might follow them. It is, however, not enough to keep the new customers happy.

Focusing on these two points concerning customer satisfaction should help Hammerkit get a solid customer base and keep it. These are of course not the only things to take into consideration but among the most important ones.

7.4 Summary of Recommendations

A continuous value stream or good communication methods will not make a company successful on their own. Customer satisfaction is however hard to achieve if these things are not in order. In order to achieve a high level of customer satisfaction a lot of different things need to be working well on their own as well as together. As stated at the beginning of this section, software manufacturing methods are on a good basis at Hammerkit. This section focused on the things the author thought were the most important for Hammerkit to consider and possibly improve in the future.

8 Personal Evaluation

This thesis began by specifying the research question *how the software manufacturing processes works at Hammerkit and how they could be improved.* The question was answered in a comprehensive way. The processes were working well and certain aspects that could be improved were identified. These were dealt with a proposition of how to go about improving them. What the thesis set out to discover was accomplished. It also managed to propose ways to improve the processes.

9 Conclusion

Software manufacturing processes are never perfect. The goal for any particular company is to improve the processes and methods continuously. The goal of this thesis was to examine these processes at Hammerkit and to find out how they are working.

The research had a wide array of references and the information was trustworthy. This thesis managed to answer the question it set out to answer. The thesis was conducted with a thorough research into the theories and the information on Hammerkit. A table of questions was formed from the theory to compare with Hammerkit. The theory and information on Hammerkit were compared to discover how the processes are working at the moment. These findings formed the basis to answering the question in table 5. These answers were analyzed and on the basis of this analysis the recommendations were made.

The research showed that the software manufacturing processes are managed well. This was to be expected from a company that specializes in software manufacturing. There is a unified way of executing a project from the beginning to the end. The employees are aware of how the project should be conducted. The structures of the products are also similar from one project to the other. The customers are involved in the process. They have the possibility to influence the end product that they will receive.

The cloud store allows Hammerkit to enter new markets and the business model is well thought through. By focusing on mass customization, Hammerkit is able to reduce the costs as well as manufacture the products faster and with better customization.

The most important things to focus on in the future should be customer satisfaction, continuous value stream and communication. Communication is a company-wide issue and concerns each and every project. It needs to work well concerning the communication methods used as well as on the personal level. Communication in Hammerkit could be improved. There is no quick-fix but rather ways to constantly keep improving it. Focusing on the value stream should also be kept in mind. It is working in Hammerkit but as with everything, it could be improved.

Customer satisfaction is the corner stone of every successful company. Happy customers are customers that are most likely going to do business with the company again. One could go as far as to state that customer satisfaction is more important than the other aspects discussed in this thesis. Customer satisfaction is a concern at Hammerkit and so should be the focus of improving it.

This thesis showed that the overall state of the manufacturing process and working methods at Hammerkit is good. As stated in the beginning of this section, things could however always be better. There is no reason to start changing things that are working well. Hammerkit should focus on customer satisfaction, communication and on even greater value production.

References

Abrahamsson Pekka, Ronkainen Jussi, Salo Outi, Warsta Juhani: 2002: Agile software development methods. VTT Publications, Espoo.

Badurdeen Fazleena, Stump Brandon 2012: Journal of Intelligent Manufacturing 2012 Volume 23, Number 1, Pages 109-124. Integrating lean and other strategies for mass customization manufacturing: a case study.

Bjørnvig Gertrud, Coplien James O., 2010:Lean Architecture for Agile Software Development. John Wiley & Sons Ltd. The Atrium, West Sussex, United Kingdom.

Blokland Wouter W.A., Curran Richard, Verhagen Wim J.C. 2009: Drivers of Customer Satisfaction in a Project-Oriented, Business-to-Business Market Environment: an Empiric Study. Global Perspective for Competitive Enterprise, Economy and Ecology, Advanced Concurrent Engineering. Part 13, pages 833-844.

Condon Dan 2002: Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales. Aspatore Books, Inc, the United States of America.

Gwaltney Jr. William A., Richardson Jared R. 2005: Ship It! A Practical Guide to Successful Software Projects. The Pragmatic Programmers LLC. Raleigh, North Carolina Dallas, Texas.

Hammerkit 2011. Available in WWW-format: http://www.Hammerkit.com/index/43 03.12.2011

Hammerkit 2012. Available in WWW-format: http://www.hammerkit.com/services 02.03.2012

Highsmith Jim 2002: Agile Software Development Ecosystems. Pearson Education, Inc. Indianapolis.

Highsmith III James A. 2000: Adaptive Software Development: A Collaborative Approach To Managing Complex Systems. Dorset House Publishing, New York.

Idsoe E.A., Skjevdal R. 2005: The Competitive Impact of Product Configurators in Mass Tailoring and Mass Customization Companies. Department of Economics and Logistics SINTEF Technology and Society. Available in WWW-format: http://www.sintef.no/project/SMARTLOG/Publikasjoner/2005/2005%20Idsoe%20Skjevdal_Competitive%20Impact.pdf 16.03.2012

Kotipizza 2012. Available in WWW-format: http://www.kotipizza.fi 20.04.2012

Krishnamurthy Nikhilsh, Saran Amitabh 2008: Building Software: A practitioner's Guide. Auerbach Publications Taylor & Francis Group, LLC. Boca Raton, New York.

Pfeifer Tilo, Schmitt Robert 2009: Success with Customer Inspiring Products – Monitoring, Assessment and Design of Perceived Product Quality. Industrial Engineering and Economics 2009, part 2, pages 117-129.

Pine II B. Joseph 1992: Mass Customization: The New Frontier in Business Competition. The Harvard Business School Press, the United States of America.

Schlesinger Richard 2010: Developing Real Life Software. Jones and Bartlett Publishers, LLC. Sudbury, Massachusetts.

Evaluation Sheet for Hammerkit

#	Statement	No	Sometimes/ almost	Yes
1	The processes and methods concerned directly with the end user			
2	The products customized for a particular customer			
3	There is a large product offering			
4	The main focus is on people, instead of methods			
5	The processes produce value all the time			
6	There is a framework from which to work			
7	There is a continuous process flow			
8	Everyone is in on the project from the very beginning			
9	Most of the communication is done face-to-face			
10	The goal is defined early on			
11	There are regular informal meetings			
12	There is a focus on customer expectations			
13	There is a focus on eliminating waste			
14	The goal is shared by all the team members			
15	The architecture gives a top level view			
16	The customers are kept informed about the progress			
17	There is a way to recognize/measure if the communication is working			
18	The employees are aware of what others are working on			